Mon. Not. R. Astron. Soc. **000**, 1–9 (2012) Printed 22 November 2012 (MN LATEX style file v2.2)

Chemical complexity in astrophysical simulations: optimization and reduction techniques

T. Grassi[1⋆], S. Bovino[2], D. Schleicher[2], F. A. Gianturco[1]
[1] *Department of Chemistry, Sapienza University of Rome, P.le A. Moro, 5, 00185 Roma*
[2] *Institut für Astrophysik Georg-August-Universität, Friederich-Hund Platz 1, 37077 Göttingen*

Accepted *****. Received *****; in original form ******

ABSTRACT
Chemistry has a key role in the evolution of the interstellar medium (ISM), so it is highly desirable to follow its evolution in numerical simulations. However, it may easily dominate the computational cost when applied to large systems. In this paper we discuss two approaches to reduce these costs: (i) based on computational strategies, and (ii) based on the properties and on the topology of the chemical network. The first methods are more robust, while the second are meant to be giving important information on the structure of large, complex networks.
To this aim we first discuss the numerical solvers for integrating the system of ordinary differential equations (ODE) associated with the chemical network. We then propose a buffer method that decreases the computational time spent in solving the ODE system. We further discuss a flux-based method that allows one to determine and then cut *on the fly* the less active reactions.
In addition we also present a topological approach for selecting the most probable species that will be active during the chemical evolution, thus gaining information on the chemical network that otherwise would be difficult to retrieve. This topological technique can also be used as an *a priori* reduction method for any size network.
We implemented these methods into a 1D Lagrangian hydrodynamical code to test their effects: both classes lead to large computational speed-ups, ranging from $\times 2$ to $\times 5$. We have also tested some hybrid approaches finding that coupling the flux method with a buffer strategy gives the best trade-off between robustness and speed-up of calculations.

Key words: astrochemistry – ISM: evolution, molecules – methods: numerical.

1 INTRODUCTION

Chemistry plays a key role in many astrophysical environments (e.g. Galli & Palla 1998, Nelson & Langer 1999, Omukai et al. 2005, Merlin & Chiosi 2007, Meijerink & Spaans 2005, Teşileanu et al. 2008, Gnedin et al. 2009, Glover et al. 2010, Yamasawa et al. 2011, Grassi et al. 2011). It regulates the cooling and the fragmentation of the interstellar gas (Jappsen et al. 2005) and it is one of the most powerful diagnostic way to relate with the observational signatures coming from the last generation of telescopes like ALMA[1] and Herschel[2]. Unfortunately tracking its evolution needs large computational resources. The complexity of the problem arises from its mathematical representation: a network of chemical reactions has its counterpart in a system of coupled ordinary differential equations (ODE) that is often stiff (i.e. with coefficient highly-varying in size). The solution of such a system may becomes too expensive computationally, especially when the number of reactions is large, or when the system is formed by many elements of fluid.

Over the years several approaches have been proposed to solve this problem: reducing the number of reactions/species making physical or chemical assumptions is one of the most used. The number and the kind of reactions being removed depends on the astrophysical environment that one wants to simulate (e.g. Nelson & Langer 1999). This method can be useful when the network is small, but it is hard to apply when a larger network is employed, leading to large errors with complex networks.

Another widely used approach is to reduce the number of non-equilibrium species (e.g. Glover et al. 2010) treating

⋆ Corresponding author: tommasograssi@gmail.com
[1] www.almaobservatory.com
[2] http://herschel.esac.esa.int/

the remaining species as being in equilibrium. This method decreases the chemical timestep and thus the computational cost, but it is problem-dependent since the selection of the non-equilibrium species depends on the environment one wants to simulate.

In this paper we discuss a set of methods that can be used to increase the feasibility of accurate simulations based on large chemical networks while also providing some key information on the structure of the network.

We introduce the problem in Sect.2, then we consider two classes of reduction strategies: (i) the first is based on methods more related to computer science and we call them "pure" computational strategies (see Sect.3), (ii) while the second set is based on the analysis of the properties and on the topology of the chemical network (Sect.4). In Sect.5 we will show tests on some of the methods previously discussed.

2 OVERVIEW OF THE PROBLEM

A chemical network is composed of a number of reactions each generally represented by $\mathrm{A}+\mathrm{B} \rightarrow \mathrm{C}+\mathrm{D}$, and the probability that a reaction occurs is related to a rate coefficient $k_i(T)$ that depends on the gas temperature T. In this framework each species evolves following an ordinary differential equation (ODE) that takes into account the reactions that form and destroy the ith species

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = \sum_{j \in form} k_j(T) n_{r1(j)} n_{r2(j)} - n_i \sum_{j \in destr} k_j(T) n_{r1(j)} \,, \quad (1)$$

where $n_{r1(j)}$ and $n_{r2(j)}$ are the number densities of the first and the second reactants of the jth reaction, and the subscript $form$ and $destr$ represents the set of the reactions that form and destroy the ith species. Note that for three-body reactions we must multiply the first part (and/or the second, depending on the reaction) of the RHS term by the number density of the third reactants, i.e. $n_{r3(j)}$. The network is then described by a system of ODEs.

From a computational point of view the problem of solving a system of ODEs resides mainly in building the RHS term of Eqn.(1), while the total number of equations has a smaller (but non-negligible) impact on the solver efficiency (Tupper 2002). To achieve this we can either remove the species, decreasing the number of ODEs and hence the dimensionality of the problem, and/or reduce the number of reactions, thus diminishing the time spent to build the RHS of Eqn.(1). Both approaches will be discussed in Sect.4.

3 PURE COMPUTATIONAL STRATEGIES

In this Section we discuss the strategies that are less related to the properties and the features of the chemical network, but allow to improve the computational performance of solving a system of ODEs aimed at tracking the chemical evolution. We first present (i) an analysis of the different solvers, then (ii) the buffering method, and finally (iii) a general discussion on standard computational techniques.

3.1 Numerical solvers

Astrochemical networks are represented by a system of stiff ODEs, which requires appropriate solvers. Due to the numerical instability of the system, one of the most used solvers is the implicit backward differencing (BDF) scheme, but depending on its implementation it can greatly affect the computational performance. A large number of solvers have been proposed over the years like Rosenbrock, Bulirsch-Stoer-Bader-Deuflhard, implicit Runge-Kutta, and Gears (Press et al. 1992), but unfortunately these methods have poor performance when applied to astrochemical networks since a large number of iterations and function evaluations are required to integrate the ODE system. A good way to solve this problem is to use the well-established ODEPACK/SUNDIALS package[3] (Hindmarsh et al. 2005), a solver suite based on GEARS and LSODE that allows to solve stiff ODE systems more efficiently. The DVODE fits our class of problems as well, but nevertheless, when the system presents a sparse Jacobian the DLSODES solver (Hindmarsh 1983), which has the capability of handling sparse matrices, is more efficient. DLSODES has been already discussed in Nejad (2005), and we have tested here the two solvers on a larger network (Wakelam & Herbst 2008) involving 452 species and more than 4000 reactions. The DVODE and the DLSODES will be tested in Sect.5.1.

3.2 Buffering

In the framework of hydrodynamical simulations one usually solves the chemical ODEs for each fluid element (i.e. each gas particle or grid cell), the number of which can easily exceed 10^6, depending on the resolution: for this reason reducing the number of calls to the solver saves a large amount of CPU-time. This allows us to take advantage of the fact that we can avoid to solve the chemical ODEs for fluid elements which have chemical conditions (e.g. temperature, species numerical densities, ...) similar to those that were already calculated, since they will show the same evolution with time. This comes from the fact that a system of ODEs is a Cauchy's problem that depends on the set of initial conditions and on the differential equations which are weighted by the values of the reaction rates.

The buffering method we propose consists of storing the already calculated chemical results. If the initial conditions of a given fluid element matches the initial conditions of a buffered element we can avoid to call the solver again, and use instead the stored evolution. In the opposite case we call the solver to evolve the fluid element and we store the initial and the final conditions in the buffer (see also Algorithm 1 discussed below).

To determine if two particles are similar we loop over the initial conditions and we check if the expression $s_i = |x_{ij} - x_i|/x_i < \epsilon$ is true, where x_{ij} is the ith initial condition of the jth particle of the buffer, and x_i is the ith initial condition of the test particle. It is worth noting that we also need a constraint on the time-steps of the two particles, namely $|\mathrm{d}t - \mathrm{d}t_j| < \epsilon_t$, where ϵ_t is the time threshold, and in the tests of Sect.5.2 we use $\epsilon_t \approx 0.1$ yr. The time threshold is

[3] computation.llnl.gov/casc/sundials/main.html

Algorithm 1 - Pseudocode for the buffering method proposed here. Note that the term *particle* is the same as *fluid element*. The aim of this psuedocode is to find the final conditions $\hat{\mathbf{x}}$ of a particle with initial conditions $\mathbf{x}$. See details in Sect.3.2.

```
 1: x ← initial conditions                          // x are the initial conditions of a given particle
 2: for (j = N, 1) do                               // loop over buffer particles (j = N → 1)
 3:     found←True                                  // set default value of found flag
 4:     if (|dt_j − dt| > ε_t) then                 // compare dt
 5:         found←False                             // set found flag to false
 6:         skip to next fluid element              // skip to the next buffer particle (next j)
 7:     end if                                      //
 8:     for (i = 1, N_c) do                         // loop over initial conditions (i)
 9:         if (|x_i − x_ji|/x_i > ε) then          // check similarity for the ith initial condition
10:             found←False                         // set found flag to false
11:             exit from initial condition loop    // breaks loop over initial conditions (i)
12:         end if                                  //
13:     end for                                     // end loop over initial conditions
14:     if (found) then                             // if particles match
15:         x̂ ← x̂_j                                 // load final conditions from buffer
16:         exit from buffer loop                   // go to the next cell (break loop j)
17:     end if                                      //
18: end for                                         // end loop over buffer particles
19: if (not found) then                             // if the particle is not in the buffer
20:     x̂ ← solver(x, dt)                           // compute final conditions with solver
21:     x̂_{N+1} = x̂                                 // store final conditions in the buffer
22:     x_{N+1} = x                                 // store inital conditions in the buffer
23:     dt_{N+1} = dt                               // store time step in the buffer
24:     N = N + 1                                   // increase the size of the buffer
25: end if                                          //
```

chosen in order to have a good approximation over the time-step and for this reason its value must be a fraction of the mean hydrodynamical time-step, depending on the desired accuracy. From our test we found that a small ϵ_t provides accurate results even with larger values of ϵ.

The total error is given by the chosen ϵ and ϵ_t, and it also depends on the chaotic behaviour of the chemical network. If the system is unstable to small perturbations of the initial parameters we must choose a very small ϵ thereby reducing the advantage of using the buffer.

In order to increase the speed-up of the method we scan the buffer from the most recent values to those stored at earlier stages, since it is more likely that the last particles stored are similar to the test particle. This increases the probability of finding a similar particle earlier during the scan, allowing us to break the cycle after a few comparisons: this increases the advantage of using this method.

Since the size N of the buffer increases while the simulation is running, we must define a maximum buffer size, namely $N_{\max}$, that is $N_{\max} = N_{\rm tot} \cdot m$ where $N_{\rm tot}$ is the total number of fluid elements employed in the simulation, and $m \geqslant 1$ (we set $m = 20$ and $N_{\rm tot} = 100$ in our tests). When the buffer is full ($N = N_{\max}$) we simply move the last $N_{\max}/2$ buffered cells at the beginning of the buffer and we set $N = N_{\max}/2$, in order to re-use the last $N_{\max}/2$ cells stored.

We provide a pseudocode (Algorithm 1) in order to clarify the above description. Note that the pseudocode refers to a given fluid element (called test element) with initial conditions $\mathbf{x}$ (line 1) that is compared to all the particles in the buffer via the loop $j = N \rightarrow 1$. The aim of this algorithm is to find the final conditions $\hat{\mathbf{x}}$ for the test particle by comparison, (line 15) or, if comparison fails, via a call to the solver (line 20). In the algorithm we first check the condition for the time-step dt (line 4) that is written in order to be false when the time-steps of the test particle and the jth buffer particle are different (line 6) to avoid useless calculations (i.e. skip to the next buffer particle). Then it loops over the set of particles to compare the initial conditions of the test particle and of the jth buffer particle (line 8). In this case we also write the conditions in order to save CPU time by exiting from the initial conditions loop (line 11). If both of these tests (time-step and initial conditions) are false, the variable *found* remains set to true (line 14), and in this case the jth buffer particle is similar to the test particle: we load then the final conditions from the buffer particle (line 15) and we break the loop over the buffer (line 16). If, after the loop over the buffer, the variable *found* (line 19) is still false we need to calculate the final conditions in the standard way (via solver, line 20), we store the new particle in the buffer (lines 21 and 22), and we increase the size of the buffer (line 24).

In the tests proposed in Sect.5, 70% of the calls to the solver are saved with a speed-up of $\times 5$ and a very small error on the chemical evolution. We also found, as expected, that the number of skipped calls to the solver decreases during the simulation, because when the shock is fully developed there are less similar fluid elements than at the beginning, but, more in general, the total number of evolved fluid elements loaded from the buffer (and hence not calculated with the solver) depends on the features of the simulated environment. Note that a large buffer ($N_{\max} \gg N_{\rm cell}$) reduces the rate of efficiency decreasing, altough a large buffer needs a non-negligible computational time to be scanned.

3.3 Other strategies

There are other strategies related to standard programming techniques that can give an additional speed-up as are the various optimization flags allowed by widely-used compilers. We don't discuss here the details which are compiler-dependent, but we remind the reader that aggressive optimization (often indicated as -O3 flag), inter-procedural optimization, and loop unrolling/vectorization can lead to significant speed-ups. We also remark that good programming practices (e.g. save divisions, avoid casting, pre-calculate mostly used expressions, ...) help the global performances of the code and should be always employed. These strategies coupled with an accurate code profiling will make the code considerably less CPU-demanding.

Evaluating the rate coefficients is another source of CPU-consumption. Often the ODE for the temperature is coupled with the others ODEs through cooling and/or heating and trough the adiabatic index (γ) that depends on the mass fractions of the species. For this reason the rate coefficients must be evaluated at each solver's step, and usually these coefficients contain intrinsic functions as exp() and log() that are expensive in terms of computer time. One way to solve this problem (while keeping the same accuracy) is to tabulate the rates and to interpolate them during the simulation when required, obtaining a speed-up up to a $\times 5$ depending on the reaction rates used.

The last solver-related issue concerns the function called by the solver that contains the differential equations surmised by Eqn.(1). These equations can be either explicitly written equation-by-equation in the code, or called by using a loop. The latter approach produces a more compact code which is easy to handle and modify, and it is faster during compilation (especially for very large networks), but unfortunately it is less efficient at runtime. In the tests proposed here we have preferred for the sake of simplicity the loop approach, even if explicitly writing the equations would give an additional speed-up.

4 THE REDUCTION METHODS

In this Section we review those reduction strategies that are more directly related to the properties and the features of the network itself. We briefly discuss the most used methods for reducing the workload associated with the chemistry, and we also introduce a new approach based on the topology of the network. The need to reduce the astrochemical networks arises from the huge computational time required for astrophysical simulations. Common practices, based on arbitrary cutting, are often used to deal with this problem, e.g. those based on chemical or physical considerations. As already stated in the Introduction, this approach is reasonable for very small networks where the complexity of the connections among the species is clear enough to decide what to remove.

There are several methods aiming at reducing the computational cost of the chemistry. A common practice is to pre-calculate the models with some one-zone chemical code in order to create a database of model evolutions, and then, when needed, to interpolate the final conditions during a large simulation. This method works fine for a small number of parameters, e.g. the temperature and the initial number densities of the species involved. When the number of species grows, the dimensionality of the database increases thus making the database huge and difficult to interpolate.

Table 1. Overview of the methods discussed in this Paper, where *d.o.f.* means degrees of freedom, *transf.* transformation, and *infos* is for informations on the structure of the chemical network. See text for further details.

Method	pros	cons
Direct	exact	very slow
Interp.	fast	few d.o.f.
ANN	fast, many d.o.f.	hard fine-tuning
User-based	fast	arbitrary
Lumping	fast	small-systems
SVD, PCA	fast	transf. overhead
Buffer	fast	efficiency decreases
Flux-based	fast, infos	DLSODES overhead
Topology	fast, infos	misses initial cond.

The latter problem can be solved with an artificial neural network (ANN) approach which works similarly to an interpolator, but can handle a larger number of free parameters (Grassi et al. 2011). Unfortunately an ANN needs a lot of fine-tuning to reproduce the results of the database, and for this reason most of the times it cannot be included in large simulations as an interpolator.

Another class of reduction methods is based on linear algebra. (i) The lumping method (Okino & Mavrovouniotis 1998) allows one to reduce the dimensionality of the problem by making some linear combination of the species involved. This method has been developed for small systems and for this reason cannot be applied to complex networks. (ii) Also the singular value decomposition (SVD) and the principal component analysis (PCA) are aimed at reducing the dimensionality of the ODE system using a transformation matrix that allows to redefine the coordinates of the space of the parameters (in our case temperature and all the numerical abundances) in order to define a new space where the dimensionality of the problem is smaller. The main drawback is that computing this transformation matrix has a non-negligible computational cost and it could easily result in a computational overhead.

The overview of these methods is presented in Tab.1, which includes their *pros* and the *cons* features in terms of usage.

We tackle this computational problem by employing instead a more robust technique known as the flux-method presented in Tupper (2002) and Grassi et al. (2012). It is an *on the fly* procedure aimed at determining the less active reactions and then reducing the number of terms in the RHS of Eqn.(1). The tests proposed in Grassi et al. (2012) showed large speed-ups ranging from $\times 2$ to $\times 10$ depending on the network employed. It is worth noting that this method has been developed for the solver DVODE, while when applied to DLSODES solver it generates an overhead since it interferes with the solver's subroutines aimed at handling the sparsity of the Jacobian (Tupper, private communication).

A new *a priori* method based on the connectivity features of a network (i.e. on topology) is therefore presented

in the next Section, and some of its key aspects and results will be discussed there.

4.1 The topology of astrochemical networks

An astrochemical network can be viewed as a *directed* graph where the vertexes (or nodes) are the species of the chemical system, while the edges (or links) are determined by reactions between species. A reaction as $A + B \rightarrow C$ is represented with three nodes (namely A, B, C) and two edges ($A \rightarrow C$ and $B \rightarrow C$). When we consider astrophysical networks the number of species/vertexes spans from a few tens to more than several hundreds, while reactions/edges can be more than five thousands. This large number of items becomes complicated to be displayed as a graph and, as a consequence, the topological features are usually not evident at first sight. Different methods have been proposed through the years to explore the properties of the networks as summarized by Jolley & Douglas (2010, 2012). A widely used method consists in measuring the degree of the nodes that comprise the network, where degree is defined as the sum of the connection of a node to its neighbourhoods. This quantity shows some interesting properties when applied to real examples. One of the most intriguing is the power-law distribution of the probability of finding a node with a given number of connections. More explicitly the probability of finding a node with a degree d decreases as $P(d) \propto d^{-\gamma}$ (where $\gamma > 0$ is a parameter), it means that finding a node with many connections is less probable that finding a less connected one.

This property is true for *scale-free* networks, that are considerably robust and stable to random node removing (e.g. Barabasi 1999). In particular when the network is robust it allows to remove some nodes without damaging the global stability of the network because the largest part of the information flows through the so called *hubs* that are the most connected elements of the system. When we deal with astrophysical networks we are interested in finding stable structures from a "noisy background" represented by the less active nodes or, as in the scheme just depicted, we want to know what node can be removed or not.

Even if astrochemical networks are not *scale-free*, since they have an exponential degree distribution instead of a power law (Jolley & Douglas 2010), we can determine which are the sub-structures that can be deleted by using some ranking techniques. The degree of a node can be a good way to determine whether a node is important or not, and then if it could be removed or not; when we decide to use the degree as a criterion we assume that it is very probable that a large part of the information will flow through very connected nodes instead of flowing through the less connected. It is evident that removing vertexes with many connections (*hubs*) could seriously damage the whole network because of their crucial role in the global network activity.

Another approach to rank the most important species is to calculate how much each node is linked to nodes that are highly connected to the rest of the network. This method results from the assumption that the most connected nodes carry the largest part of the global network information and in particular we choose that being the neighbour of a very connected node increases the probability of becoming active during the network evolution. This fact can become more clear when we take a social network as an example: in this case a person (a node) can easily reach critical information if it is connected to a few individuals that massively participate to the information exchange (*hubs*), rather than being in contact with a large number of poorly connected persons.

We call this "second degree" and it can be represented as the sum of the degrees of the nodes close to a given node (i.e. its neighbours). For the ith node is

$$^{(2)}d_i = \sum_{j \in N_i} d_j \,, \tag{2}$$

where $^{(2)}d_i$ is the second degree of the ith node, N_i is the set of the neighbours of the ith node, and d_j is the degree of the node j.

In this way the second degree becomes a parameter that is more powerful than the degree itself, because it takes into account not only the activity of a given node, but also the activity of its close neighbours. In principle we can also calculate the nth degree by iteratively applying the idea above for $n \rightarrow \infty$

$$^{(n)}d_i = \sum_{j \in N_i} {}^{(n-1)}d_j \,. \tag{3}$$

This is similar to the Google PageRank (Page et al. 1999) that assigns a numerical weighting to each web page determining its relative importance. Both algorithms have an iterative approach aimed at finding the probability distribution that represents the likelihood that a given species participates to the global network activity, or a person randomly clicking on various links will arrive at a particular page.

In this first analysis, to compute the second degree we consider the graph as *undirected*, since to determine the activity of a given node we have to include the edges that leave that node, which represent the destruction reactions, and the edges pointing at that node, being reactions which form the species represented by the given node. Under this assumption the activity of the node is determined by both the reactions that form and destroy the corresponding species, hence the need to consider the graph as undirected.

There are three approaches to calculate the second degree and the difference between them is determined by the weight assumed during the calculation of the degree d_i in the equation

$$d_i = \sum_{j \in N_i} w_j d_j, \tag{4}$$

where w_j is the weight. The first method to obtain d_i is simply (i) to count the number of connections that reach or leave a node assuming $w_j = 1$. This method allows to compute the second degree without considering the time evolution, but it neglects whether the reactions carrying information to the node are important or not. (ii) To improve on that choice, one can change the weight to $w_j = k_{ij}$ where k_{ij} is the rate coefficient of the reaction that involves the ith node as product and the jth node as a reactant. This is more accurate than the first approach, but it cannot be calculated once and for all since k_{ij} is temperature-dependent. This method also ignores the fact that at a given time a reaction could not be active because the abundance of the reactants is zero. (iii) The latter issue can be avoided by

considering $w_j = F_{ir}$ where F_{ij} is the flux of a reaction calculated as $F_{ij} = k_{ij}\, n_j \prod_r n_r$ where n_j is the abundance of the neighbour reacting species, and the product runs over the other reactants excluding n_j. This calculation can be only performed at runtime, because one needs to know the abundances of the species during the simulation, and moreover, the rate coefficient k_{ij} depends on the temperature, which changes during the evolution of the model.

In this paper we shall show the results only for the first method that can be used as a good estimator to determine the most important species, allowing to eliminate from the network the less active ones before calculation. The other two methods have more general validity and accuracy but require evaluation during runtime, which is a significant drawback from the computational standpoint.

Once we have ranked the species in the network we can choose a criterion to remove the less important ones, and to choose this we want to make sure that we do not remove species with high abundances. To cope with this risk we introduce a second degree threshold d_n based on species densities

$$^{(2)}d_n = \min\left[{}^{(2)}d_i\right] \forall\, i \,|\, n_i > n_t\,. \tag{5}$$

that depends on the lowest of the second degrees $^{(2)}d_i$ of the species with a number density n_i greater than a user-defined density threshold n_t. We also define a second threshold d_f to keep important species when the first threshold can be excessively constraining (e.g. initial monoatomic gas), namely

$$^{(2)}d_f = f \cdot \max\left[{}^{(2)}d_i\right]\,. \tag{6}$$

which is determined by a certain fraction $f \leqslant 1$ of the maximum second degree.

We decide to cut the species that do not satisfy both criteria at the same time, hence the final threshold becomes

$$^{(2)}d_t = \min\left[{}^{(2)}d_n, {}^{(2)}d_f\right]\,, \tag{7}$$

which gives the final criteria $^{(2)}d_i > {}^{(2)}d_t$ employed to decide whether or not the ith species is kept in our calculation. In the tests we present in Sect.5.3 we use $f = 0.6$ and $n_t = 10^{-5}$.

5 TEST CASES

We propose here tests involving a selection of the aforementioned methods, employing the OSU reactions database as in Wakelam & Herbst (2008) - without PAHs - coupled to a simple 1D Lagrangian code aimed at simulating a Sedov-like gas shock (Bodenheimer et al. 2006). The gas is composed of a set of spherical shells representing an inner region with initial conditions $T_{\rm in} = 10^4$ K and $\rho_{\rm in} = 10^{-20}$ g/cm^3, and another set of shells for the outer region with $T_{\rm out} = 10$ K and $\rho_{\rm out} = 10^{-21}$ g/cm^3 and radius $R = 1pc$. The number of the inner shells is $N_{\rm in} = 40$, while the number of the outer shells is $N_{\rm in} = 60$, giving $N_{\rm tot} = 100$. Note the results found for the tests presented in this Section are still valid for larger systems and also for systems based on an Eulerian approach, and 3D hydrodynamics.

The initial abundances are the same as the EA2 model of Wakelam & Herbst (2008) (i.e. solar metallicity initial conditions) and we let the system evolve for 10^6 yr. All the tests presented in this paper employ the DLSODES solver with an absolute and relative tolerance of 10^{-40} and 10^{-12} respectively (which are accurate enough compared to the values of the number densities usually employed in ISM simulation), and an internally-generated Jacobian (option MF=222, more details in Hindmarsh 1983). Note also that the code has largely been optimized and compiled with Intel® Fortran Composer XE 2011 Update 6 using a standard set of optimization flags.

It is important to remark that since the scope of these tests is to determine the computational efficiency of the methods proposed, we use the OSU database as it is (osu_01_2007, 4431 reactions and 452 species) and, moreover, we do not include any cooling nor heating. Under the latter assumption the hydrodynamics of the shock is not affected by the chemical evolution. However, the tests discussed here represent - computationally speaking - a typical ISM scenario with a large chemical network. We plan to describe the coupling between hydrodynamics, chemistry and cooling together with reduction methods in a forthcoming paper.

The methods are all compared with the *full* evolution, i.e. without any reduction. To show the accuracy of the different methods we have plotted the abundances of a given species in all the gas shells for each reduction method compared to the same shell in the *full* model. More in detail the plots represent the accuracy in reproducing the chemical behaviour of the gas during the evolution of the shock in all the shells for a given species. If a method reproduces exactly the chemical evolution of the *full* model all the points will lie along a straight line (i.e. $y = x$), while the distance from the line increases if the method fails to reproduce the original model. These plots represents the dispersion of the values obtained for a given method ($n_{\rm method}$) compared to the values of the full model $n_{\rm full}$. We have also included two lines representing the magnitude of fluctuations of one order of magnitude in each direction (i.e. $y = 10\,x$ and $y = 0.1\,x$ labelled +odm and -odm respectively).

We found that coupling together *buffer* and *flux* methods gives the best results both in terms of robustness and of speed-up.

5.1 Comparison of solvers

The characteristics of the chemical network play a key role to determine what is the best numerical solver. In this framework we propose a 1D shock model test mainly focused on Jacobian sparsity, using DVODE and DLSODES solvers (Hindmarsh 1983; Hindmarsh et al. 2005). It is important to remark that employing an internally generated or a user-provided Jacobian will affect the performance of the solver, since building the Jacobian has a non-negligible cost for the solver; however for the sake of simplicity in all our tests we use this latter method.

The Jacobian associated with the network of the OSU database is extremely sparse ($\sim 94\%$), which is very common for astrochemical networks. For this reason in the shock test framework the numerical solver DLSODES outperforms the more general DVODE solver achieving approximately a $\times 100$ speed-up.

5.2 Buffer method

We test the *buffer* method with two thresholds, namely $\epsilon = 10^{-1}$ and $\epsilon = 10^{-5}$, the latter being a finer approximation. We found good results especially for the 10^{-5} threshold which reproduces the full model almost exactly, but, as expected, when increasing the threshold value ($\epsilon = 10^{-1}$) the method starts to be less accurate in reproducing the original data, but the error remains always below one order of magnitude (see Fig.1 and Fig.2).

5.3 Topological method

The *topological* method reproduces the *full* model with a good accuracy even if some dispersion is observed for the C^+ and H species (Fig.1, top and Fig.2, bottom). The threshold we chose selects 2369 reactions over 4431, hence it removes almost 50% of the network connections. As stated in the previous Section, the reduction based on the topological method has a probabilistic meaning and its best property lies in the capability of giving information on which species/reactions are important or not. For instance, our tests show that the most important *hubs* (the most connected species) are, in that order, free electrons, H, H_2, He^+, C, H^+, O and C^+, as expected. This topological approach finds that species like Cl, Mg, Fe, and their ions, but also MgH, and HF and many C-chains species, are not so important within this network, thus many reactions involving He^+ and C^+ are neglected. The latter can explain the fluctuation of C^+ species as shown in Fig.1. We want to underline once again that, unlike *flux* or *buffer*, the topological method is mainly aimed at giving *a priori* chemical information on the network rather than strongly reducing the number of the reactions, or the species. A more robust reduction technique based on topology has already been discussed: using the flux as the weight in Eqn.(4) will provide a more accurate analysis. However, this method is not *a priori*, and for this reason it could generate a large overhead when the reduction is less effective. Moreover, from a computational point of view, this technique is closer to the *flux* method, since the weights are exactly the fluxes and the two methods are largely comparable. The only advantage is the amount of topological information provided during the system evolution. Due to these considerations, it seems likely that flux-based reduction techniques are preferable when a robust reduction method is required, even if the error associated with the *topology* method is generally less than one order of magnitude. Depending on the desired accuracy, the method can nevertheless be useful in some applications.

5.4 The flux-based method

For the *flux* method presented here we assume that the length of the hydrodynamical time-step is equal to the length of the macro-step, which is a good approximation since temperature and total density will remain constant during a hydrodynamical time-step. Under this assumption the evaluation of the fluxes is made at the beginning of each hydrodynamical time-step for each gas shell. This allows one to couple the DLSODES solver with the flux method. The results (Fig.1 and Fig.2) are in good agreement with the full solution and can be further improved using a more accurate definition of macro-step (see Grassi et al. 2012 for further details).

Table 2. Normalized CPU-time measured for different reduction methods. See text for further details.

Method	$t_{\rm CPU}$
Full	1.00
Topology	0.37
Buffer (-1)	0.40
Buffer (-5)	0.61
Flux	0.50
Flux+Buffer	0.24
Flux+Buffer+Topology	0.17

Table 3. Speed-up overview for the methods discussed here. Note that the values listed here refer to the ISM model presented in this paper. One can find different speed-ups depending on the chemical network employed, the numerical framework, and some other parameters. However, these values are representative for the methods discussed and can be used as reference.

Method	Speed-up
Solvers	$\times 100$
Reduction methods	up to $\times 10$
Buffering	up to $\times 5$
Rates tabulation	$\times 5$

5.5 Hybrid methods

The last method discussed here is a mixing between *buffer* ($\epsilon = 10^{-5}$), *flux*, and *topology*. The accuracy of this hybrid method is determined by the least accurate approach as we can see in Fig.1 (top), where topology fails, and also in Fig.2 (bottom) where topology is not as accurate as the other methods.

The normalized CPU-times of the different methods are reported in Table 2: as expected hybrid methods are the fastest, but coupling *flux* and *buffer* shows a better accuracy than the coupling of *flux*, *buffer*, and *topology*. We also obtain a large speed-up for the coarsest of the two *buffer* methods, while the one with $\epsilon = 10^{-5}$ has the smallest speed-up, although it has the best accuracy among the methods proposed. Finally, the *topology* shows good speed-up with less accurate results. It is worth noticing that the latter reduction method gives important information about the chemical network due to its topological approach, and that to really see the effects of such a cut on the global hydrodynamical evolutions we need a test which couples the chemistry to the dynamics via heating and cooling, and via photon diffusion.

6 CONCLUSIONS

In this Paper we have discussed some techniques aimed at reducing the computational cost of including a chemical network into astrophysical codes that simulate the evolution of the ISM. We have divided the methods into two classes, namely the *"pure" computational strategies* that are more related to computer science, and the *reduction methods* that

are based on the properties and on the structure of the chemical network. In the first class we discussed the solvers employed, the buffering method, and a brief review on other strategies, while in the second class we described the flux method and introduced a topology-based method.

To test the different methods we employed a 1D hydrodynamical Sedov-like shock test with a standard OSU (osu_01_2007) chemical network, which is a large chemical network (> 4000 reactions). The methods tested are *buffer*, *topology*, and *flux*, including some hybrid methods that couple the previous ones. These methods reproduce the results of the *full* model with good speed-ups, except for the topology-based approach which has a larger error (compared to the other methods), since it is mainly devoted to rank by importance the different species rather than providing a robust reduction technique as the *flux* method. Nevertheless, the results provided by this method suggest that the topological approach can lead to an efficient reduction based on the features of the chemical network, which gives at the same time some critical information about the global properties of the interconnections between the various species included in the ISM model.

Finally, we found that the most robust method among those examined turns out to be the coupling between the *flux* approach and the *buffer* technique, and this hybrid method also achieves one of the best speed-ups (almost $\times 5$) as suggested by our tests. The fastest shock simulation is obtained, as expected, by using *flux*, *buffer*, and *topology* all together, but the *topology* affects the global behaviour of this hybrid method producing results that are less accurate than *flux* and *buffer* methods and their hybrid.

A final overview of the speed-ups obtained from the various methods presented in this paper is given in Table 3.

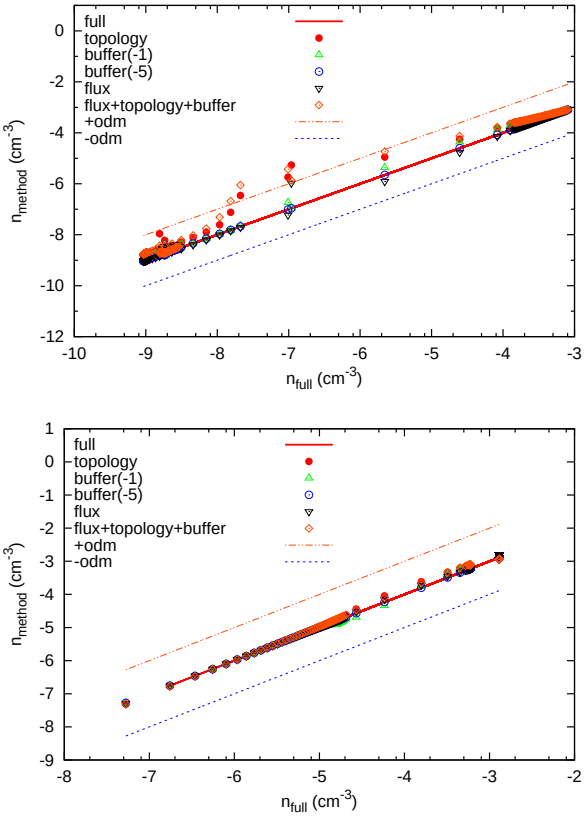

Figure 1. Comparison of the *full* model with the different reduction methods for C^+ (top) and CO (bottom). As indicated in the key the methods are *topology* (dots), *buffer* with $\epsilon = 10^{-1}$ (triangles up) and $\epsilon = 10^{-5}$ (open circles), *flux* (triangles down), and hybrid (diamonds). For colour figure see the online version.

ACKNOWLEDGEMENTS

T.G. acknowledges the financial support from the CINECA and S.B. and D.R.G.S. thank for funding through the DFG priority program 'The Physics of the Interstellar Medium' (projects SCHL 1964/1-1). D.R.G.S. thanks for funding via the SFB 963/1 on "Astrophysical Flow Instabilities and Turbulence".

REFERENCES

Barabasi A. L., 1999, Physica A Statistical Mechanics and its Applications, 272, 173
Bodenheimer P., Laughlin G., Rozyczka M., Yorke H., 2006, Numerical Methods in Astrophysics: An Introduction. Series in Astronomy and Astrophysics, Taylor & Francis
Galli D., Palla F., 1998, A&A, 335, 403
Glover S. C. O., Federrath C., Mac Low M.-M., Klessen R. S., 2010, MNRAS, 404, 2
Gnedin N. Y., Tassis K., Kravtsov A. V., 2009, ApJ, 697, 55
Grassi T., Bovino S., Gianturco F. A., Baiocchi P., Merlin E., 2012, MNRAS, 425, 1332
Grassi T., Krstic P., Merlin E., Buonomo U., Piovan L., Chiosi C., 2011, A&A, 533, A123
Grassi T., Merlin E., Piovan L., Buonomo U., Chiosi C., 2011, ArXiv/1103.0509
Hindmarsh A. C., 1983, IMACS Transactions on Scientific Computation, 1, 55
Hindmarsh A. C., Brown P. N., Grant K. E., Lee S. L., Serban R., Shumaker D. E., Woodward C. S., 2005, ACM Trans. Math. Softw., 31, 363
Jappsen A.-K., Klessen R. S., Larson R. B., Li Y., Mac Low M.-M., 2005, A&A, 435, 611
Jolley C., Douglas T., 2012, Astrobiology, 12, 29
Jolley C. C., Douglas T., 2010, ApJ, 722, 1921
Meijerink R., Spaans M., 2005, A&A, 436, 397
Merlin E., Chiosi C., 2007, A&A, 473, 733
Nejad L. A. M., 2005, Ap&SS, 299, 1
Nelson R. P., Langer W. D., 1999, ApJ, 524, 923
Okino M. S., Mavrovouniotis M. L., 1998, Chemical Reviews, 98, 391
Omukai K., Tsuribe T., Schneider R., Ferrara A., 2005, ApJ, 626, 627
Page L., Brin S., Motwani R., Winograd T., 1999, Technical Report 1999-66, The PageRank Citation Ranking: Bringing Order to the Web., `http://ilpubs.stanford.edu:8090/422/`. Stanford InfoLab
Press W. H., Teukolsky S. A., Vetterling W. T., Flannery

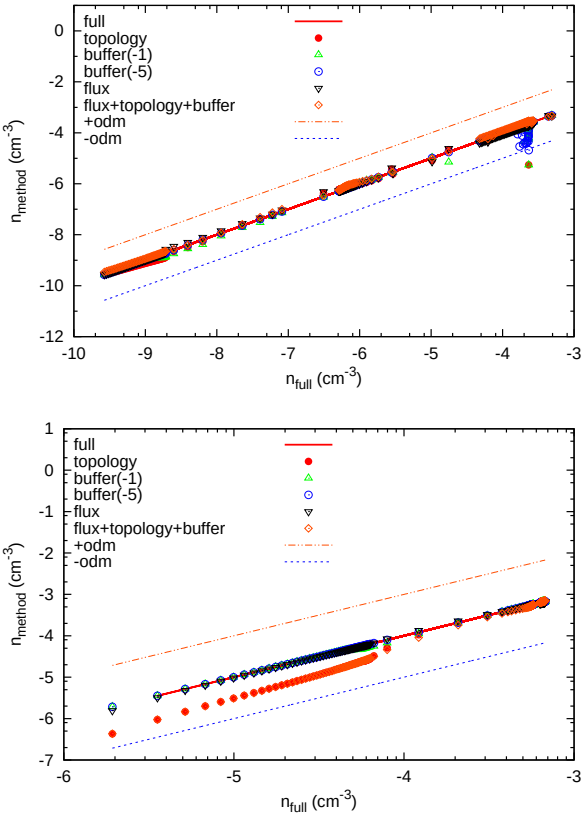

Figure 2. Comparison of the *full* model with the different reduction methods for OH (top) and H (bottom). As indicated in the key the methods are *topology* (dots), *buffer* with $\epsilon = 10^{-1}$ (triangles up) and $\epsilon = 10^{-5}$ (open circles), *flux* (triangles down), and hybrid (diamonds). For colour figure see the online version.

B. P., 1992, Numerical recipes in FORTRAN. The art of scientific computing

Teşileanu O., Mignone A., Massaglia S., 2008, A&A, 488, 429

Tupper P. F., 2002, Bit Numerical Mathematics, 42, 447

Wakelam V., Herbst E., 2008, ApJ, 680, 371

Yamasawa D., Habe A., Kozasa T., Nozawa T., Hirashita H., Umeda H., Nomoto K., 2011, ApJ, 735, 44

This paper has been typeset from a TEX/ LATEX file prepared by the author.